复旦大学腾飞书院先导课程
程序设计分册

前　言

在过去的数十年，除非你一直与世隔绝，否则就不可能不受到信息革命的影响。我们身处技术演进史上的计算机时代，从艾伦·图灵开始，到克劳德·香农、冯·诺依曼、恩格尔巴特，再到蒂姆·伯纳斯·李，如果没有他们的启迪，就不会有计算机和互联网的飞速发展。

在我们生活的这个时代，你会发现有这样一群人，他们对世界的影响越来越大：比尔·盖茨创立了微软，让计算机更容易被我们平常人所使用；乔布斯创立了苹果公司，它们的产品改变着我们的日常生活；谢尔盖·布林和拉里·佩奇创立了谷歌，使得信息的获取变得前所未有的容易；马克·扎克伯格创立了 Facebook，改变了人与人之间的社会关系；伊隆·马斯克创立了 SpaceX 公司，挑战航天事业……他们为什么会创造奇迹？他们有一个共同的特点：在少年时代都酷爱计算机编程。计算机编程究竟具有怎样的非凡魔力？计算机编程是否给他们带来与常人不同的思维或思考方式？是否是计算机编程为他们开启了不一样的人生道路？

计算机从被发明的那一天开始，就是来帮助我们提高学习和工作的效率的。计算机编程或程序设计是给出解决特定问题程序的过程，是软件构造活动中的重要组成部分。程序设计往往以某种程序设计语言为工具，给出这种语言下的程序。程序设计过程应当包括分析、设计、编码、测试、排错等不同阶段。程序设计也是人工智能、计算机生物、虚拟现实等计算机前沿领域的基础。使用计算机编程，你可以设计好玩的游戏，帮助处理日常的事务，轻松解决数学难题。

在这本讲义中，我们为即将进入大学学习的新生们提供两周的 C 语言程序设计基础能力训练教程，在暑期两周为新学期助力！亲爱的同学们，开启我们的编程之旅，来解决这些有趣的问题吧！

本《程序设计分册》由复旦大学计算机科学技术学院的周雅倩老师、信息科学与工程学院的郑达安老师编写。

目　　录

第一天　编程平台的下载与安装

“工欲善其事，必先利其器。”要学好 C 语言编程，首先要有一个足够权威的编程平台。我们推荐的平台是 DEV C++。

§1.1　下载 DEV C++

DEV C++ 是一个开源软件，我们可以在 Sourceforge 上下载到（可能有点慢，但这是比较安全的做法）。具体网址是：http://sourceforge.net/projects/orwelldevcpp/?source=directory进入网站后，点 Download 就好啦。

§1.2　安装 DEV C++

下载后打开安装文件，静候其 Loading 到 100%，就可以开始安装了。由于安装包只有英文版的，所以一开始的语言我们就只好选择 English 啦。不过请放心，安装之后可以选择中文界面的。

点击 I Agree 按钮，进入选择安装类型的界面。目前来讲，Full 和 Custom 模式是一样的，而 Safe 模式不仅安装全套功能，还会删除计算机上一些 DEV C++ 的老数据，可以防止运行过程中出现错误。Full、Minimal、Safe、Custom 模式安装所需空间其实相差很小，所以可以选择 Full、Custom 或者 Safe 模式，体验 DEV C++ 的所有功能。

接下来，选择安装路径，一般可以按默认路径安装，不必改动。如果读者的系统盘空间实在不足，可以选择安装在其它位置，毕竟所需空间有三百多 M。

接下来的安装过程结束后，点击 Finish 完成安装。就可以开始运行 DEV C++ 了。第一次运行需要进行设置。在 Set your language 一栏可以选择“简体中文/Chinese”以及其它语言。点击 Next 进行下一步设置。

接下是选择主题，读者可以选择自己喜欢的字体、颜色、图标，如果想看起来“逼格”高一点，就给自己选一个黑底或者蓝底的配色吧。这一步过后，初始设置步骤就完成了。

接下来会进入 DEV C++ 的界面。点击左上角菜单栏“文件”中的“新建”或者菜单栏下面的空白文件图标，可以新建 C 语言源文件。由于我们目前只是编点小程序，所以可以只新建“源文件”，而不用新建“项目”。以后我们需要调试的时候，最好就还是新建一个项目啦。

如果是新建项目，那么会跳出一个对话框。我们应该选择 Console Application，并勾选“C 项目”，然后给自己的项目起一个喜欢的名字。

好啦，相信大家已经迫不及待想翻开下一章，开始奇妙的 C 语言之旅啦。

第二天　Hello World 程序

§2.1　Hello World

和其它语言一样，C 语言作为通用计算机语言，是人与计算机交流的一种工具。任何程序都是由主程序、若干子程序和数据结构组成的。主程序叫main函数，子程序叫函数。（本节读者只需了解 C 程序的组成就可以了，细节将在后面各章节中详细介绍。）

接下来我们来看一个初学者的典例：Hello world 程序。

```
1 #include<stdio.h>  /*在程序中使用标准函数库中的输入输出函数*/
2 void main()
3 {
4     printf("Hello world\n");
5 }
```

例 2.1　Hello World 程序

本程序的功能是在显示器上显示一行字符：Hello world。下面对本程序逐行说明：

(1) 第一行#include是编译预处理指令，是在编译前将stdio.h这个文件里的系统定义好的函数包含到我们写的程序中，参与编译。/*　*/是注释，注释由/*开始，*/结束。注释可插入到程序的任何地方，内容可为英文或中文，目的是使程序便于理解。

(2) 第二行是main函数说明语句。（void表示函数的返回值类型为空类型；返回括号中可以放函数的参数，此函数无参数，所以括号内无内容。）

(3) 第三行是main函数体左括号，开始符。

(4) 第四行是格式输出语句。printf函数是系统定义好的格式化输出函数，用于向终端（显示器等）输出字符，调用格式为：printf("格式化字符串", 参量表)。\n是换行符，转义字符的一种。

(5) 第五行是 main 函数体右括号，结束符。

代码输入过程中，注意用英文输入，括号、引号和分号等标点符号使用半角输入（即() " " ; ）不要输成全角的符号（即 （） “” ；）。输入完成后，点击运行菜单栏中的编译运行或运行，也可以使用快捷键F10或F11；如果运行成功，那么恭喜你已经开始了编程之旅。如果有报错，可以仔细核对一下你的代码，看看是不是拼写有错。

§2.2　程序框架

C 程序是由函数构成的。一个 C 程序包括一个main函数和若干其它函数。其中main函数是主函数，程序从它开始执行，是必有的；其它函数可以是编程者自定义的函数，也可以是已定义好的库函数。每个函数都有函数名和函数体，函数体用一对花括号{}括起来。

C 程序的基本结构可表示为：

```
#include<文件名>
返回值类型 函数名()    /*自定义函数*/
{
    函数体
}
int main()  /*int是另一种函数返回类型，有些为空类型，如上节中的void*/
{
    函数体
}
```

如同我们说话一样，函数体中包含许多语句，每个语句结束后有一个分号（;）。Hello world 程序中的main函数非常简单，只含一个语句。接下来我们看一个稍微复杂一点的例子：输入两个数（以空格隔开），输出两个数之和。

```
#include<stdio.h>
int main()
{
    int a, b, sum;/*定义三个变量*/
    scanf("%d%d",&a, &b); /*输入两个数*/
    sum = a + b; /*将a与b相加的值赋给sum变量*/
    printf("%d + %d = %d\n", a, b, sum); /*输出结果*/
    return 0;
}
```

例 2.2 求两数之和

程序分析：

(1) 第 4 行int表示变量的类型为整型（详细的数据类型将在后面介绍），三个变量名分别为a，b，sum。变量间用逗号“，”隔开；

(2) 第 5 行scanf和printf类似，是格式输入函数，%d表示以十进制格式输入两个数，分别放入&a，&b地址指定的两个存储单元中，做为变量a，b的值；

(3) 第 7 行格式输出语句中%d表示以十进制输出三个数a,b,sum。

每日一练

(1) 以下选项哪个是 C 语言中的包含头文件写法：

A. include<stdio.h>　　B. #include stdio.h

C. #include<stdio.h>　　D. #include[stdio.h]

(2) 主函数很重要，来看看你会了没有

在右侧编辑器中第 2 行输入主函数名称：

```
#include<stdio.h>
_____________/*这里输入我们的主函数哦*/
{
```

```
4       printf("C程序中一定是从我开始的");
5       return 0;
6   }
```

(3) 输入 Hello world 程序，并运行程序。

(4)（选做）编写程序，输入两个数（以空格隔开），输出两个数之和。

第三天　基本数据类型及其运算

虽然现代计算机可应用于各行各业，但计算机最主要的应用还在于它的计算能力。完成一次计算至少包含两方面的内容：要运算的数据和运算的方法。在高级语言中，形式不一的数据用不同的数据类型来区分，将所有可能的数据分成基本的和结构化的两类。一种高级程序设计语言首先要确定有哪些基本类型的数据，提供哪些结构化的构造方法来构造复杂形式的数据，然后为每种形式的数据分别提供一组运算方法。C 语言能在字符、整数、浮点数等基本类型基础上，按结构化的层次构造方法，构造数组、结构等各种结构化的数据类型，基于它结构化的程序控制结构，可以编写结构非常良好的程序。

§3.1　基本数据类型

在高级语言中，形式不一的数据用不同的数据类型来区分，在 C 语言中，基本数据类型有三种，分别是整数、浮点数和字符。虽然整数和浮点数都是无穷多的，但是计算机只能用有限的二进制位表示不同的数据，所以计算机只能表示有限的整数和浮点数，并且浮点数是近似表示。而字符型数据的内部表示也是二进制串，也就是字符的编码（ASCII 码），而非字符本身。下面我们来介绍一下基本数据类型。

整型数据： 不带小数点和指数符号的数据，按数值范围分类。经常使用的是`int`类型，通常4 B/8 B（32 bit/64 bit）表示（C 语言的`int`类型与开发环境有关，可能是32位，也可能是64位，现在大多数开发环境都是64位，B：Byte 内存字节，1 个 Byte 有 8 个bit），其中32位能储存的数据的数值范围为：−2 147 483 648至2 147 483 647（即 -2^{31} 至 $2^{31}-1$）。

浮点型数据： 带有小数点或指数符号的数据，按精度分为三种。单精度型`float`，具有6至7位精度；双精度型`double`，具有15至17位精度；长双精度型`long double`，具有更高位精度。这里的精度指的是数据有效数字位数。例：

0.000 000 000 000 006

6E−15，6e−15（表示6×10^{-15}）

浮点型数据的格式建议为以上两种：整数、小数或整数 E(e) 指数。

字符型数据： 表示一个字符，占用1 B，表示的数值为0至255或−128至127，用`char`表示。例如：

```
char ch ='A';
```

这个语句定义了一个`char`类型的变量ch，并赋予初值`'A'`（ASCII 编码为65），字符在内存中是以它 ASCII 编码的形式储存的，实际的编码值取决于计算机环境，其中最常见的为美国标准信息交换码（ASCII）。

`char`类型变量具有两重性：可以把它解释为一个字符，也可以把它解释为一个整数（ASCII 编码）。一个字符型的数在参加运算时是以它 ASCII 编码的形式出现的。例如：

```
char letter1 = 'C'; /*ASCII编码为67*/
char letter2 = letter1 + 3;
    /*运行后以整型数形式输出letter2 是70，以字符型式输出是 'F'*/
```

§3.2 变量

如同自然语言一样，程序语言也有基本符号，再由基本符号按一定的构词规则构成基本词汇，最后按语言的语法由基本词汇构成的语句序列组成源程序。C 语言的基本符号包括数字、英文字母、下划线和其它用于构成特殊符号的字符集合。基于这些基本符号构成字面形式常量（比如100，3.14）、特殊符号（比如运算符）、关键字（比如int）、和标识符这四类基本词汇。标识符是用来命名变量、函数等程序对象的。在 C 语言中，一个合理的标识符由英文字母或下划线开头，后接若干个字母、下划线和数字符组成的字符列。在程序运行过程中，值不能改变或者不允许改变的数据对象称为常量。变量是在程序运行过程中可以设置和改变值的数据对象。

变量在存在期间，在内存中占据一定数量的存储单元，用于存放变量的值。程序用变量与现实世界中的数据对象对应，与变量相关的概念有变量名、变量数据类型、变量在内存中的地址、变量的值等。程序通过变量定义引入变量，变量定义的一般形式如下：

```
类型 标识符列表
```

其中，标识符列表由用逗号分隔的一个或多个标识符组成，每个标识符作为一个变量的名称。例如：

```
int i, j, sum;          /*定义3个int型变量*/
char ch;                /*定义1个char型变量*/
double d;               /*定义1个double型变量*/
```

变量定义时，还可以为变量指定初值，也就是变量初始化。例如：

```
int i = 5;
```

这个语句定义了一个 int 类型的变量 i，并且赋予初始值 5，这里 5 就是一个字面常量。

§3.3 数据运算

数据运算由操作数和运算符组成，C 语言提供了非常丰富的运算符来描述各种数据的运算，这里我们介绍算术、赋值、复合赋值和自增减，这四种常用的运算符。

基本的算术运算： +，-，*（乘），/（除），%（取模，除法运算后的余数），例：

```
1 int radius = 4;
2 float area = 0.0;
3 float perimeter = 0.0;
4 area = 3.1415926*radius*radius; /*结果是50.265461*/
5 perimeter = 3.1415926*2*radius; /*结果是25.132740*/
```

除法运算符/略微特殊一点，整数运算总是得到整数结果，例如表达式1/2结果不是0.5，而是0，“整数除法”返回被除数和除数相除后的整数部分。但只要算式中带有一个浮点数，其结果就会是浮点数。

取模运算符%是对除法运算的补充，它提供了一种在整数除法运算后获取其余数的方式，通常来说(int)a%(int)b == a - (int)(a/b)*b，但对于被操作数是负数时，余数的符号由所使用的编译器决定。

最后两行中=是赋值语句。

赋值运算符： 即=，计算赋值运算符右边的算术表达式，结果会储存在赋值运算符左边的变量中，这与代数方程中的=不太一样，例：

```
int i = 6;
i = i + 1;
```

变量i初始化为6，表达式i+1等于7，这个结果储存回i，所以其效果是i的值增加了1。

建议使用括号控制复杂表达式的执行顺序。

复合赋值运算符： 包括+=，-=，*=，/=，%=在内的OP=赋值运算符（OP表示 Operator，即运算符），它的一般形式为：lhs = lhs OP (rhs);。例如，

```
feet %= feet_per_yard;
```

等价于

```
feet = feet % feet_per_yard;
```

注意lhs = lhs OP (rhs)式中的“()”很重要，例：

```
x *= y + 1;
```

它代表的是：

```
x = x * (y + 1);
```

而不是：

```
x = x * y + 1;
```

自增减运算符： 即++和--，当自增自减运算符放在变量名之前，则先执行自增或自减后，再计算整道表达式的返回值；当自增自减运算符放在变量名之后，则先计算整道表达式的返回值，再执行自增或自减。例：

```
int count = 6, i = 0;
i++;
count--;
```

执行i++;语句后，i递增1，变为1；执行count--;语句后，count递减1，变为5。

§3.4 输入输出

§3.4.1 输出函数

C 语言中输出可以使用printf()和putchar()函数。其中printf()可做任意输出，而putchar()仅可以输出单个字符。printf()函数的使用形式为

```
printf("字符串和类型说明符",变量);
```

putchar()函数的使用形式为putchar(c);，其中c可为一个被单引号引起来的字符（如'A'），一个介于0到127之间（包括0和127）的十进制整数（会被识别为对应字符的 ASCII 编码，输出该编码对应字符）或字符型变量，例：

```
int salary = 1500;
printf("My salary is %d.",salary);
```

屏幕上会显示：

```
My salary is 1500.
```

以上语句中printf("")内的信息会被输出，按照顺序My salary is 得到了输出，当遇到格式说明符时，格式说明符将会被变量所存储的数值代替，这意味着当编译器遇到%时将认为你要输出变量中储存的数据（用\%可输出'%'这个字符）。

```
char a = 'A';
putchar('A');
putchar(65);
putchar(a);
```

屏幕上会显示：

```
AAA
```

§3.4.2 输入函数

C 语言中输入可以使用scanf()和getchar()函数,scanf()可以输入任何类型的数据,而getchar()，顾名思义只能获取单个字符。scanf()函数的使用形式为：

```
scanf("格式说明符",待输入赋值的变量的地址);
```

而getchar()函数括号内为空，例：

```
int i;
scanf("%d", &i);
```

此时你在键盘上输入8，则变量i被赋值为8。在上个语句中，%d是格式说明符，它表明即将接受到的数据为整数类型，&为寻址运算符，&i为变量i的地址（内存地址），scanf()函数可以利用变量的地址，允许键盘输入的数据存入变量。其中的原理涉及到指针的知识，传入地址是为了用指针修改变量的值，若不用传入地址的方式，则没办法将输入的内容存储到变量中去。有兴趣的同学可以自己了解一下 C 语言的指针类型。

```
char a;
a = getchar();  /*getchar函数返回输入的字符 */
/*C语言中的函数像数学的函数一样，也会得到一个值，这个值我们称为返回值，
   同时，用运算符进行运算得到的值我们也称为返回值。关于函数的内容详见第
   十章*/
```

此时你在键盘上输入A（并按下回车），则变量a被赋值为'A'。

§3.4.3 输入输出格式说明

printf()和scanf()函数对输入输出数据的格式需要通过使用格式字符区分，常见的格式字符及说明如表3.1。

表 3.1 输入输出格式字符表

格式字符	说明
%d	以十进制形式输出整形数据
%o	以无符号八进制形式输出整形数据
%X，%x	以无符号十六进制形式输出整形数据（%X时字母大写）
%c	以字符型式输出，只输出一个字符
%s	输出字符串
%f	以小数形式输出浮点数（默认六位小数）

在之前的printf("My salary is %d.",salary);语句中，因为salary是整数，所以输入输出时使用%d。

§3.5 例子：四则运算

```
1  #include <stdio.h>     /*为了调用scanf()和printf()函数*/
2  int main()
3  {
4      float math, algebra, physics, programming;
5      float GPA;
6      printf("请输入你的各科绩点(数学分析,线性代数,大学物理,程序设计):\n"
           );
7      scanf("%f%f%f%f", &math, &algebra, &physics, &programming);
8      GPA = 5.0*math + 3.0*algebra + 4.0*physics + 4.0*programming;
9      GPA /= 5.0 + 3.0 + 4.0 + 4.0;
10     printf("你的基础课绩点为%f",GPA);
11     return 0;
12 }
```

例 3.1 四则运算

每日一练

(1) 基本数据类型、赋值、输入输出格式

任务：小李今年 18 岁，身高 172 厘米，体重 82.5 公斤，小李是否属于肥胖呢？我们如何用程序描述这个情形呢？

在右边编辑器中，在第 4、5、6、7 行中将变量的类型补全、赋值并在 8，9，10、11 行补上输出格式，最后运行程序。

```
1  #include <stdio.h>
```

```
int main()
{
    _______ age = _______;
    _______ height = _____;
    _______ weight = ______;
    _______ isfat = 'y';
    printf("年龄：%______岁\n", age);
    printf("身高：%______cm\n", height);
    printf("体重：%______kg\n", weight);
    printf("是否肥胖：%_____\n", isfat);
    return 0;
}
```

(2) x经过两次加运算结果为 2，而每次输出的结果分别为：

```
x=0
x=2
```

但是以下代码好像写错了，请帮助把编辑器中的代码改正吧。

```
#include <stdio.h>
int main()
{
    int x = 0;
    printf ("x=%d\n", x++);
    printf ("x=%d\n", x++);
    return 0;
}
```

(3) int a = 3, b = 4;计算a+b%(a+b)/a+a*b-b的运算结果为

A. 算式结果为：11　　B. 算式结果为：12

C. 算式结果为：14　　D. 算式结果为：15

(4) 在代码编辑器中的第 7 行和第 9 行分别填入实现代码，使得运行结果为

```
X = 20
Y = 20
```

```
#include <stdio.h>
int main()
{
    int x = 10;
    int y = 10;
    //使用简单赋值语句实现x乘以2。
    ------------
    //使用复合赋值语句实现y乘以2。
    ------------
```

```
10      printf("x=%d\n", x);
11      printf("y=%d\n", y);
12      return 0;
13  }
```

(5) 编写程序：给定一个球体的半径为6，试计算出该球体的体积。（提示：球体积的计算公式 $V = \frac{4}{3}\pi R^3$）

(6) 编写程序：输入 2 个整数，输出这 2 个整数的积、商和余数的程序。

(7) 编写程序：给定字母'a'和'A'，输出它们对应的 ASCII 码值。

第四天　选择结构

不论是在解决数学问题的过程中还是平时的生活中，我们常常会碰到一些问题需要分情况解决，比如在超市，我们买了东西得付钱了才能出大门，要是没买东西就可以直接离开超市了。用“如果-那么-否则”的形式来表达就是“如果我在超市买了东西，那么就得付钱，否则就不用付钱”。在 C 语言程序设计的过程中，我们也会有这样需要根据当前情况而进行不同的计算的情形。用于实现这样的计算的程序结构称为**选择结构**。在了解选择结构之前我们有必要了解 C 语言中的关系表达式和逻辑表达式。

§4.1　关系表达式

含有**关系运算符**的表达式称为**关系表达式**。C 语言中，有 6 个关系运算符，它们分别是：<（小于），<=（小于等于），>（大于），>=（大于等于），==（等于），!=（不等于）。

其中小于和大于运算符与通常数学上的表达形式相同。小于等于、大于等于、等于及不等于的形式与数学上有一定的不同，需要记忆。需要引起我们注意的是，要注意区分等于运算符（==）和前面章节中提到过的赋值运算符（=）。

关系表达式对关系运算符左右两侧的值进行比较。如果比较运算成立，那么关系表达式的值即为1，如果比较运算不成立，那么关系表达式的值即为0。

我们来看一个例子。

```c
#include<stdio.h>
int main(){
    int a = 1, b = 1, c = 2;
    printf("%d\n", a > b);
    printf("%d\n", a >= b);
    printf("%d\n", a + b == c);
    return 0;
}
```

例 4.1　关系表达式示例

在上述 C 程序中，由于a与b相等，所以表达式“a > b”不成立；而“a >= b”成立；又由于a + b的值与c相等，所以表达式“a + b == c”成立。故输出结果应为：

```
0
1
1
```

§4.2 逻辑表达式

含有**逻辑运算符**的表达式称为**逻辑表达式**。C 语言中有 3 个逻辑运算符，它们分别是：&&（逻辑与）、||（逻辑或）、!（逻辑非）。

&&（逻辑与）是一个双目运算符，当且仅当其两侧表达式的值都非0时，其运算结果值为1，否则其值为0。对于表达式“a && b”，有如下真值表：

a \ b	0	非0
0	0	0
非0	0	1

||（逻辑与）是一个双目运算符，当且仅当其两侧表达式的值都为0时，其运算结果值为0，否则其值为1。对于表达式“a || b”，有如下真值表：

a \ b	0	非0
0	0	1
非0	1	1

!（逻辑非）是一个单目运算符，当其后表达式值非0时，其运算结果值为0；当其后表达式为0时，其运算结果值为1。对于表达式“! a”，有如下真值表。

a	! a
0	1
1	0

我们来看一个例子：

```
1 #include<stdio.h>
2 int main(){
3     int a = 1, b = 1;
4     printf("%d %d %d\n", a && b, a || b, ! a);
5     printf("%d\n", a > b && a >= b);
6     printf("%d\n", a > b || a >= b);
7     return 0;
8 }
```

例 4.2 逻辑表达式

根据第4.1节中关系表达式值的判断，表达式“a > b”的值为0，表达式“a >= b”的值为1，所以本程序输出结果为：

```
1 1 0
0
1
```

§4.3 if…else…选择结构语句

选择结构中较为常用的即为与“如果-那么-否则”的形式类似的“if…else…”语句，称为**两路条件选择结构**。

“if…else…”语句的一般形式为：

```
if (表达式)
    语句1;
else
    语句2;
```

其具体执行过程为：

(1) 计算表达式（往往是关系表达式或逻辑表达式）的值；

(2) 若表达式的结果非0，执行语句 1；若表达式的结果为0，执行语句 2。

容易发现，“if”和“else”分别对应“如果”和“否则”，语句 1 和语句 2 则分别对应“如果”和“否则”之后需要执行的具体过程。

有必要注意的是，当语句 1 和语句 2 都只能是一个 C 语句，如果需要执行多个语句，则需要用{ }使得这些语句构成一个复合语句。

在选择结构中，有时我们只需要执行“if”所对应的语句 1，那么else以及语句 2 是可以省略的，这个称为if语句。

当需要区分的情况不止两种时，还可以将“if…else…”语句嵌套使用。例如：

```
if (x == y)
    printf(“%d”, 0);
else if (x > y)
    printf(“%d”, 1);
else printf(“%d”, -1);
```

再来看一个例子：

```
#include<stdio.h>
#include<math.h>
int main(){
    float a, b ,c;
    float s,area;
    scanf("%f%f%f", &a, &b, &c);
    s = (a + b + c) / 2.0;
    if (a + b > c && b + c > a && c + a > b)
        area = sqrt(s * (s - a) * (s - b) * (s - c));
    else
        area = 0.0;
    printf("%f",area);
    return 0;
}
```

例 4.3 if…else…语句

本例中，如果a、b、c能够构成一个三角形的三条边，则我们利用熟知的海伦公式求出该三角形面积，否则令面积值为0。

§4.4 选择结构应用实例

本节中，我们来看一个经典实例——输入三个整数，输出它们中的最大数。

解题思路：任取三个整数中的其中一个作为临时的最大数，再将其余两数分别与临时最大数比较，如果一数大于临时最大数，则临时最大数取该数的值，最后输出最终的最大数。

程序代码如下：

```
#include<stdio.h>
int main(){
    int a, b, c, max;
    printf("输入三个整数: ");
    scanf("%d%d%d", &a, &b, &c);
    max = a;
    if (b > max)
        max = b;
    if (c > max)
        max = c;
    printf("三数中的最大数是: %d", max);
    return 0;
}
```

例 4.4 输出最大数

每日一练

(1) 学校入选篮球队的资格条件为身必须大于等于180 cm，小李身高185 cm，那么小李能入选吗？在代码编辑中补全代码，判断小李能不能进入校篮球队。

运行结果应为：

```
恭喜，小李可以参加校篮球队
```

```
#include <stdio.h>
int main()
{
    int height = 185;
    //补全此处所有代码
    return 0;
}
```

(2) 编写程序，输入三个整数，输出它们的最小数。

(3) 编写程序，输入三个整数，输出它们构成的三角形中最大角的余弦值。

提示：如不能构成三角形，应输出错误信息；根据高中所学余弦定理计算。

(4) 编写程序，输入考生得分，根据复旦大学本科生学籍管理规定输出考生绩点及等级。

等级	绩点	百分制参考标准
A	4.0	90-100
A-	3.7	85-89
B+	3.3	82-84
B	3.0	78-81
B-	2.7	75-77
C+	2.3	71-74
C	2.0	66-70
C-	1.7	62-65
D	1.3	60-61
F	0	59 及 59 以下（不及格）

第五天　循环结构

通常，计算机要处理一系列数据，会出现重复计算。控制重复计算过程可用循环结构。循环结构用于描述在某个条件成立时，重复执行某个计算或操作。循环结构主要有控制循环的条件和一个重复的循环体组成。

假设我们要求 $1+2+3+\cdots+100$ 的值，我们可以直观的先计算 $1+2$ 得到 3，然后计算 $3+3$ 得到 6，然后计算 $6+4$ 得到 10，……，最后计算 $4950+100$ 得到 5050。当然我们知道 $1,2,\cdots,n$ 是个等差数列，所以它们的和也可以用公式 $\frac{n(n+1)}{2}$ 直接计算出来。如果我们要求 $12+22+32+\cdots+1002$ 的值呢？要计算自然对数底 e 和圆周率 π 这两个常数呢？

$$e = 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \cdots$$
$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{9} - \frac{4}{11} + \cdots$$

以上的这两个常数虽然有公式，但是需要一项一项地计算，来达到预定的精度。这些问题我们都可以编写程序，用重复计算的方法来解决。

以上的这些问题，用程序中的循环过程，却是非常方便解决。下面我们以等差数列求和以及质数判断来讲述 C 语言中的两种循环结构语句：while语句和for语句。

§5.1　while语句

这里我们首先使用重复计算的方法，用while语句来实现求sum= $1+2+3+\cdots+100$ 的值得程序。代码如下：

```
1  #include <stdio.h>
2  int main(){
3      int i=1, sum=0;
4      while(i <= 100){
5          sum += i;
6          i++;
7      }
8      printf("1+2+3+...+100 = %d\n",sum);
9      return 0;
10 }
```

例 5.1　用while语句，求sum=$1+2+3+\cdots+100$ 的值

在上例的代码中，i <= 100是循环条件，也称为循环表达式。循环体是一个复合语句，该复合语句由两条简单语句构成。while语句的一般形式是：

```
while(表达式)
    语句
```

表达式的计算结果如果是非 0，则执行语句，否则结束循环。语句可以是简单句，复合句，也可以是空语句。只有一个分号的语句称为空语句。

§5.2 for语句

质数又称素数，指在一个大于 1 的自然数中，除了 1 和此整数自身外，不能被其它自然数整除的数。质数是与合数相对立的两个概念，二者构成了数论当中最基础的定义之一。基于质数定义的基础之上而建立的问题有很多世界级的难题，如哥德巴赫猜想等。检查一个正整数 N 是否为素数，最简单的方法就是试除法，将该数 N 用小于等于根号 N 的所有素数去试除，若均无法整除，N 则为素数。2002 年，印度人 M. Agrawal、N. Kayal 以及 N. Saxena 提出了 AKS 质数测试算法，证明了可以在多项式时间内检验是否为素数。

这里我们使用试除法用for语句来实现判断整数n是否为质数的程序。代码如下：

```
1  #include <stdio.h>
2  int main(){
3      int n,i;
4      printf("请输入一个大于3的整数: ");
5      scanf("%d", &n);
6      for(i = 2; i*i <= n; i++){
7          if(n%i == 0)
8              break;
9      }
10     if(i*i <= n)
11         printf("%d是合数\n",n);
12     else
13         printf("%d是质数\n",n);
14     return 0;
15 }
```

例 5.2 用 for 语句，判断 n 是否为质数

在上例的代码的for语句中，i = 2给变量i赋初值，i*i < n是循环条件（即i小于等于 $\sqrt{n}$），i++修正变量i的值。循环体是一个if语句，if语句中，若条件成立，则执行break语句。执行break语句会退出当前循环。for语句的一般形式是：

```
for(变量赋初值表达式; 循环条件表达式; 变量修正表达式)
    语句
```

for语句的执行过程是：

(1) 计算变量赋初值表达式。
(2) 计算并测试循环条件表达式，若其值非0，转步骤 (3)；否则，结束循环。
(3) 执行循环体语句。

(4) 计算变量修正表达式。

(5) 转向步骤 (2)。

for语句的一般形式也可以等价地用以下形式的while语句来表达：

```
变量赋初值表达式;
while (循环条件表达式){
    语句
    变量修正表达式;
}
```

§5.3 嵌套的循环结构

中国古代算书《张丘建算经》中有一道著名的百鸡问题：公鸡每只值 5 文钱，母鸡每只值 3 文钱，而 3 只小鸡值 1 文钱。用 100 文钱买 100 只鸡，问：这 100 只鸡中，公鸡、母鸡和小鸡各有多少只？这个问题流传很广，解法很多，但从现代数学观点来看，实际上是一个求不定方程整数解的问题。

当循环结构的循环体中又包含循环结构时，循环结构就出现嵌套的形式。这里我们用两层嵌套循环来解决百鸡问题。

解题思路：采用穷举法，令变量cocks、hens、chicks分别表示公鸡、母鸡和小鸡的只数，对三种鸡所有可能的只数作循环测试，就能找到解。

```
1  #include <stdio.h>
2  int main(){
3      int cocks,hens,chicks;
4      for(cocks=0;cocks<=20;cocks++){
5          for(hens=0;hens<=33;hens++){
6              chicks=(100-5*cocks-3*hens)*3;
7              if(cocks>=0&&cocks+hens+chicks==100)
8                  printf("公鸡%d只，母鸡%d只，小鸡%d只\n",cocks,hens,
                       chicks);
9          }
10     }
11     return 0;
12 }
```

例 5.3 用嵌套循环，解百鸡问题

在上例的代码中，外层的循环控制变量是cocks，内层的循环控制变量是hens。在执行过程中，它们的变化规律是对外层的每个cocks，内层的hens就要经历一遍完整的变化。该代码的执行将输出：

```
公鸡0只，母鸡25只，小鸡75只
公鸡4只，母鸡18只，小鸡78只
公鸡8只，母鸡11只，小鸡81只
公鸡12只，母鸡4只，小鸡84只
```

§5.4 算法 1：穷举法

穷举法的基本思想是根据题目的部分条件确定答案的大致范围，并在此范围内对所有可能的情况逐一验证，直到全部情况验证完毕。若某个情况验证符合题目的全部条件，则为本问题的一个解；若全部情况验证后都不符合题目的全部条件，则本题无解。穷举法也称为枚举法。

用穷举法解题时，就是按照某种方式列举问题候选答案的过程。针对问题的数据类型而言，常用的列举方法一有如下三种：

(1) 顺序列举：是指答案范围内的各种情况很容易与自然数对应甚至就是自然数，可以按自然数的变化顺序去列举。

(2) 排列列举：有时答案的数据形式是一组数的排列，列举出所有答案所在范围内的排列，为排列列举。

(3) 组合列举：当答案的数据形式为一些元素的组合时，往往需要用组合列举。组合是无序的。

例如上面的百鸡问题和质数问题都是使用穷举法来解决的。

每日一练

(1) 用while循环实现100以内所有整数之和。

在代码编辑器中的第 6、9 行输入相应代码。运行结果应为：

```
100以内所有整数之和为：5050
```

```
1  #include <stdio.h>
2  int main()
3  {
4      int i,sum=0;
5      i=1;
6      while(________) //循环条件应该是什么呢？
7      {
8          sum=sum+i;
9          ________     //这里是不是应该改变循环变量的值
10     }
11     printf("100以内所有整数之和为：%d\n", sum);
12     return 0;
13 }
```

(2) 体验一下 for 循环，实现一个 10 以内的整数之和的小程序。

在代码编辑器中的第 7、9 行补全代码。运行结果应为：

```
10以内整数的和为：55
```

```
1  #include <stdio.h>
2  int main()
3  {
4      // 定义变量sum, num
```

```
5       int sum, num;
6       sum = 0;
7       for(num = 0; _____; ______) //for循环条件与num的变化值
8       {
9           _____ //计算每增加一个数字的结果sum
10      }
11      printf("10以内整数的和为: %d", sum);
12      return 0;
13  }
```

(3) 编写程序解决百鸡问题。

(4) 编写程序判断输入的n是否为质数。

(5) 编写程序，利用公式

$$\pi = 4 - \frac{4}{3} + \frac{4}{5} - \frac{4}{7} + \frac{4}{11} + \cdots$$

计算 π 的值至小数点后 4 位。

第六天　编程与调试

§6.1　程序运行原理

计算机由硬件和软件组成。硬件包括中央处理器（CPU）、内存储器（RAM、ROM）、外存储器（磁盘、光盘等）和输入输出设备（键盘、鼠标、扫描仪、显示器等）等设备。软件包括系统软件（例如操作系统、语言处理程序、系统实用程序等）和应用软件（建立在系统软件的基础上开发及运行的软件等）。

接下来我们来了解一下程序运行的原理，看看 C 语言是经过哪几个步骤转变成一个可以运行的程序的。

(1) 预编译，执行的是将头文件放到程序的开头、实现程序中的宏替换等过程，形成一份完整的源代码。

(2) 编译，编译过程中，C 语言被翻译成更加贴近计算机底层的汇编语言。

(3) 汇编，由汇编器将转换成汇编语言的文件转换为“可重定向目标程序”.o文件。

(4) 链接，将.o 文件与另外一些重要文件（比如一些.lib 文件）的链接，经一系列步骤获得可执行文件。

当我们运行该程序的时候，操作系统会将程序装载进内存里，通过调度 CPU、管理进程等实现对程序的运行。

§6.2　程序调试简介

所谓“人非圣贤，孰能无过。”在编程过程中我们总会遇到各种各样的小错误，甚至是大错误，在读程序的过程中，可能也难以发现。下面我们来看一个简单的例子。我们先新建一个 C 语言工程（方法详见第一章），然后写一个程序，输出 2^{32}：

```
1 #include<stdio.h>
2 int main(){
3     int i, k=1;
4     for(i=0;i<32;i++){
5         k*=2;
6     }
7     printf("2的32次方等于%d\n",k);
8     return 0;
9 }
```

例 6.1　错误的程序示例

运行程序，发现输出的是 0，不是 2^{32}。粗看程序，并不觉得有什么错误，现在我们使用调试功能。

点击“调试”按钮，我们会发现，这跟我们平时运行没有多大的区别。

现在我们添加断点，即让程序在适当的位置停下来。添加断点有不只一种方法，比如在某一行单击右键，选择“切换断点”；或者在将光标放在该行，按下键盘上的F4键；而最简单的方法是在编辑器左边显示行号的地方，用左键单击一个行号，则这一行就会成为断点。现在我们将第 5 行设为断点。

添加断点之后还需要添加查看，这样，我们可以在程序一行一行运行的同时，查看某一些变量的值，甚至是整个数组各个元素的值。现在我们点击“调试”按钮旁边的“添加查看”，将i、k添加到查看当中去。现在我们可以开始调试了。程序会在第 5 行停下来。多次点击“下一步”之后，我们可以发现，在i = 31的时候，k变成负数，i = 32的时候，k就变成了0。

现在我们不难想到，这是int类型能表示的范围不足造成的。现在只需要把k从int类型改成long int类型，在用printf输出的时候用%ld占位符，就可以输出 2^{32} 了。

这只是调试功能简单的应用，在学完数组、函数之后，我们可以尝试调试界面的其它按钮，使用其它功能。

每日一练

(1) 请调试以下程序：

```
#include<stdio.h>;
void main()
{
    float f,c;
    printf("请输入华氏温度:");
    scanf("%f",f);
    c = 5/9 * (f - 32);
    printf("对应的摄氏温度:%.2f\n",c)
}
```

第七天　数组

用基本数据类型可以解决所有问题吗？例如，对学生成绩进行查询，如果要查询张三的成绩，就要在全校学生中一个一个寻找，一直到找到为止；但如果将全校学生分成多个班级，每个学生都是班级中的一员，就可以在某某班级中寻找张三，这样就更加方便快速了。因此我们引入一个储存数据的新概念——数组。

数组，顾名思义是把相同类型的一系列数据统一编制到某一个组别中。这样我们就可以通过数组名 + 索引号简单快捷地储存、操作大量数据。在接下来的一章中，让我们更全面、详细地了解数组。

§7.1　数组的定义与初始化

§7.1.1　数组的概念

程序设计中为处理方便，把具有相同类型（如字符型，整型等）的数据按序组织成一个集合，该集合称为数组。这些数据称为数组的元素。数组元素是组成数组的基本单位且元素的数据类型相同。每个元素按其存储顺序对应一个从0开始的顺序编号，称为元素的下标。

§7.1.2　数组的声明

数组声明的一般形式为

```
类型说明符　标识符[整型常量表达式]
```

如：`int a[4]`。其中：

(1) 类型说明符是数组元素的类型。例中`int`表示元素类型为整型。

(2) 标识符是数组名。

(3) 数组元素的下标从0开始。例如数组a的 4 个元素分别为：`a[0]`, `a[1]`, `a[2]`, `a[3]`。

(4) “[]”中的整型常量表达式表示数组元素个数。

(5) 常量表达式包括常量和符号常量，不能包含变量，即使变量已有值也不可以。例如：`int n=1; int a[n];`如此定义也是错误的。

程序在引用数组元素的值之前，必须先为数组元素设置初始值。数组元素的初始值可以输入也可以用赋值语句来设置。为了程序表达简便，可以在数组定义时就给出它元素的初值。如果定义数组时未初始化，之后也没有相应的赋值操作，那么输出的值是随机的。

§7.1.3　数组初始化

数组定义时就给出它的元素的初值，称数组初始化。

初始化有下面几种情况：

(1) 数组定义时，顺序列出数组全部元素的初值。例如：

```
int a[4]={5,7,9,1};
```

(2) 定义时只给数组的前面一部分元素设置初值，未被明确设置初值的元素自动被设为0。例如：

```
int a[4]={5,7,9};
```

如此初始化之后，a[3]=0。

(3) 当全部元素都明确设置初值时，可以不指定数组长度。例如：

```
int a[]={5,7,9,1};
```

在初始化数组的同时，声明了数组a[]有 4 个元素。

§7.2 数组的输入和输出

方法：用循环实现对各个元素的输入、输出。

步骤：(1) 定义数组；(2) 用scanf函数输入；(3) 用printf函数输出。

```
#include<stdio.h>
int main(){
    int x[10], i;
    for(i = 0; i < 10 ; i++)
        scanf("%d", &x[i]);
    for(i = 0; i < 10; i++)
        printf("%d\t", x[i]);
    return 0;
}
```

例 7.1 一维数组的输入

§7.3 数组的查找、插入和删除

§7.3.1 数组的查找

数组的查找即在已知数组a中查找值为key的元素的下标。

解题思路：从第 1 个元素开始，顺序查找至第 $n-1$ 个元素。如果数组中存在值为key的元素，程序找到后结束；如果不存在值为key的元素，程序遍历完整个数组后结束。

此程序的部分代码如下：

```
for(i = 0; i < n; i++)  /*从数组第一个元素开始考察至末元素*/
    if(a[i] == key)  break; /*找到立即终止循环*/
/* i<n表示找到; i==n表示没找到* /
```

§7.3.2 数组元素的插入

数组元素的插入即在数组某个下标为k的位置插入一个新的元素x。

解题思路：设数组a原来有 n 个元素，要求在k下标位置插入一个新元素x。如果直接将a[k]替换成x，原有的元素就会丢失，所以必须将a[k]至a[n-1]分别顺序后移一个位置，然后就可将a[k]赋值为x。插入后，数组的元素个数增加 1。

此程序的部分代码如下：

```
for(i = n-1 ; i >= k ; i--) /*从a[n-1]开始到a[k]逐一后移*/
    a[i+1] = a[i];
a[k] = x;
n++;     /*数组元素增加一个*/
```

请同学们思考：为什么不能从a[k]开始后移直到a[n-1]呢？这样输出的结果又是什么呢？[①]

§7.3.3 数组元素的删除

数组元素的删除即在数组中将某个下标为k的元素x删除。

解题思路：设数组a有 n 个元素，删除a[k]后，为填补空缺，必须将元素a[k+1]至a[n-1]依次前移一个位置。删除后，数组元素个数减 1。

此程序的部分代码如下：

```
for(i = k+1; i < n; i++)     /* 从a[k+1]开始至a[n-1]逐一前移*/
    a[i-1] = a[i]];
n--;          /* 数组元素减少一个*/
```

§7.4 算法 2：二分查找

如果我们需要在一个很大的数组里查找一个元素，在最坏的情况下，我们需要将整个数组都搜索一遍。那么有没有什么更好的办法呢？如果数组是已经排好序的，那么二分法将使查找元素的速度大大加快。

二分法简介：对于已排好序的数组a，将n个元素分为个数大致相同的两半，取a[n/2]与欲查找的key作比较，若a[n/2]==key，说明找到，若a[n/2]<key，则在数组a右半部区间继续查找（假设数组元素呈升序排列），若a[n/2]>key，则在数组a的左半部继续搜索key。

有了二分法，每次将查找区间分为两半，都可以把必须搜索的范围减少一半，因此，在数据量很大时，二分查找的速度比顺序查找要快得多。

做法：

(1) 假设数组a按已从小到大顺序存放。

(2) 对任意a[i]至a[j]（$i \leqslant j$），根据它们值的有序性，试探下标为m=(i+j)/2的元素a[m]。a[m]与key比较有三种结果：

- key==a[m];，找到，查找结束；
- key<a[m];，下一轮查找区间为 $[i, m-1]$；
- key>a[m];，下一轮查找区间为 $[m+1, j]$。

① 答：因为这将使a[k+1]至a[n]都被设为a[k]。

(3) 直至 $j < i$ 时，$[i, j]$ 即一个空集，查找结束，表示数组a中没有值为key的元素。

此程序的部分代码如下：

```
i=0;  j=n-1; /*初始查找区间*/
while(i <= j){
    m = (i+j)/2;
    if(key == a[m]) break;
    else if (key < a[m])
        j = m-1;
    else
        i = m+1;
}
/*找到时，i<=j；没找到时，i>j*/
```

§7.5 算法 3：选择排序

在终端输入 n 个数，输出输入序列的一个有序排列，即重新排列这 n 个数，使其按从小到大（或从大到小）排列。

解题思路： 使用选择排序的方法，每一次从待排序的数据元素中选出最小（或最大）的一个元素，存放在序列的起始位置，直到全部待排序的数据元素排完，得到一个新的有序数组。选择排序（Selection sort）是一种简单直观的排序算法。例如：数组a = {3, 5, 2, 1, 4}，排序的过程如下所示：

- 初始序列　　：[3　5　2　1　4]（3 和 2 交换，2 和 1 交换）
- 第一趟排序结果：1　[5　3　2　4]（5 和 3 交换，3 和 2 交换）
- 第二趟排序结果：1　2　[5　3　4]（5 和 3 交换）
- 第三趟排序结果：1　2　3　[5　4]（5 和 4 交换）
- 第四趟排序结果：1　2　3　4　5

代码如下：

```
#include <stdio.h>
#define N 5
int main( ){
    int i, j, x;
    int a[N];
    for(i = 0; i < N; i++)/*输入N个整数*/
        scanf("%d", &a[i]);
    for(i = 0;i < N; i++){
        for(j = i+1; j < N; j++){
            if(a[i]>a[j]){/*a[i]和a[j]的值交换*/
                x = a[i];
                a[i] = a[j];
                a[j] = x;
            }
```

```
15          }
16      }
17      for(i = 0; i < N;i++)
18          printf("%d\t", a[i]);
19      return 0;
20  }
```

例 7.2 选择排序

每日一练

(1) 在如下程序的第 7 行-10 行之间补全数组遍历输出代码。
运行结果应为

```
0,1,2,3,4,5,6,7,8,9
```

```
1  #include <stdio.h>
2  int main()
3  {
4  int arr[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9};
5  //补全代码实现对数组arr的遍历输出
6  //可以采用你自己喜欢的循环方式
7
8
9
10
11 return 0;
12 }
```

(2) 替换指定数组中的最大元素为指定整数（此处设为 1）。
在代码编辑器的划线处填写相应代码。运行结果应为：

```
10 1 3 12 22
```

```
1  #include <stdio.h>
2  int main()
3  {
4      int arr[] = {10,41,3,12,22};
5      int value = 1   //指定整数
6      int max = arr[0];
7      int index = 0;
8      int i;
9      for(i = 1; i < 5; i++)
10     {
11         if(arr[i] > max)
```

```
12      {
13      ____________________//将数组中较大的数赋值给max
14      index = i; //记录当前索引
15      arr[index] = value;
16      for(i=0;i<5;i++)
17      {
18          printf("%d ",arr[i]);
19      }
20      return 0;
21  }
```

(3) 编写程序：定义一个长度为 10 的数组，输入 10 个自然数存入数组，然后进行由小到大的排序，最后将排完序的数组元素值输出。

第八天　程序设置

C 语言的编译系统有预编译功能。所谓预编译，就是在编译程序之前，先对预编译命令进行解释，生成新的、完整的源文件的过程。预编译命令以“#”号开始，句尾不加“;”。比如我们前面接触到的#include <stdio.h>，就是将stdio.h这个头文件中的代码（包含对许多库函数的声明）加入到源文件中，使我们能够使用一些具有强大功能的库函数。

§8.1　宏定义

下面我们将介绍宏定义这种预编译命令。

§8.1.1　不带参数的宏定义

如果现在你去年做出了一个班级管理系统，其中用到许多的数组，每个数组都有 100 个元素，又有许多循环、函数等用到了数组的大小，而学校今年想用你的程序管理整个学校的学生，你将会在将“100”改成“5000”甚至更大的数字以容纳全校学生的过程中花费大量的时间，那么有没有一个更为简便的方式可以省去这个麻烦呢？

C 语言的预编译命令中提供了宏定义的功能，可以在编译之前将所有不在引号内的宏名替换为规定好的字符序列。

相信同学们已经想到上述问题的解决方法了。如果我们在一开始写程序的时候就将常用的常量定义为宏，在使用时用宏名而不是直接使用这些常量。则要修改这些常量的时候，只需要修改宏定义就好了，这将大大提高我们修改程序的效率。而且定义宏名可以赋予一个常量意义（特别是数字常量），便于程序的阅读与修改，比如将-1用EOF代替（表示 End of file），则可以让程序员在处理文件过程中应当使用-1的时候更容易理解这个常量的意义。

定义宏的方式是

```
#define 宏名 字符序列
```

其中，宏名不能有空格，而字符序列可以有空格。比如我们用#define MAXN 100就可以用MAXN表示100了，在定义和遍历数组中，我们常用这个小技巧，让我们修改数组大小时只需修改宏定义，而不用从整个程序中寻找“100”（况且找到的100也不一定全是表示该数组的大小的），并将想要修改的“100”改成我们想要的数字。而且MAXN这个宏名也有助于程序员认识到这个常量的意义，即数组的大小。

让我们来看一下示例程序:

```
1 #include <stdio.h>
2 #define MAXN 100
3 #define PI 3.14
```

```
4  /*生成一张半径与圆周长、圆面积的对应表，储存在数组里并输出*/
5  int main() {
6      int i;
7      double peri[MAXN],area[MAXN];
8      for(i=0;i<MAXN;i++){
9          peri[i]=2*PI*i;
10         area[i]=PI*i*i;
11     }
12     for(i=0;i<MAXN;i++){
13         printf("半径为%5d的圆周长为:%12.3lf, 面积为: %12.3lf\n",i,peri[
               i],area[i]);
14     }
15     return 0;
16 }
```

例 8.1 生成一张半径与圆周长、圆面积的对应表，储存在数组里并输出

运行后，便能够输出半径为 0 至 99 的圆的周长和面积。接下来，如果我们想要提高半径的范围到 999，只需要将MAXN从100改为1000即可；如果想提高计算的精度，可以将PI从3.14改为3.1415926等等。在更大型的程序中，宏定义的优势将更大（想象一下，在一百万行的代码里找到所有100并将一部分改成1000是多么恐怖的事情！）。

当然，宏定义不仅可以替换数字，也可以替换其它字符序列。如果我们有一个程序有许许多多的地方要输出换行符，我们可以用#define ENDL printf("\n")，使得我们想换行的时候写起来更为方便。

细心的同学会发现示例中所有宏名都是由大写字母组成，这并不是硬性规定，但是为了与变量等区分开来，C 语言程序员一般会使用大写字母来当作宏名。

§8.2 命令行参数

本章我们来介绍 Windows 命令行在编程中的简单运用。主要介绍输入输出的重定向功能与文件的比较。

如果我们在测试一个有一大串输入的程序，每一次运行都要在终端输入一大堆的数据，是不是很麻烦呢？Windows 的命令行中有输入输出重定向的功能，将文件作为printf的输出和scanf的输入，可以帮助我们解决这个问题。首先让我们按键盘上的“开始”键和“R”键，或者在开始菜单中找到“运行”，并在弹出来的对话框中输入cmd，打开 Windows 的命令行。然后先尝试着打开一个文件。假设我们将一个叫Main.exe的可执行文件储存在 D 盘的C_Programming文件夹下。先输入D:，进入 D 盘，然后用cd C_Programming命令进入C_Programming文件夹。现在就可以输入main或者main.exe打开我们的可执行文件了。解释一下，打开与当前目录不同的硬盘分区要用先进入该分区，然后用cd命令进入子目录，然后直接输入可执行文件的名称以执行该文件。

接下来让我们尝试一下输出的重定向。在D:\C_Programming目录下，我们用main.exe > out.txt命令运行main.exe（假设它输出Hello world.）大家会发现屏幕上没有输出，但是目录下多了一个文件out.txt，打开后，其中写着Hello world.。这是因为，我们用>out.txt将标准输出重定向到out.txt这个文件里。

现在我们换一个例子，学习输入输出的重定向与文件的比较。

现在老师给你出了一道题，用 C 语言对 100 个整数进行排序并输出，在文件in.txt中给了你 100 个测试数据，在文件ans.txt中给了你正确答案。在测试过程中，如果手动输入，则每次我们都要输入 100 个数据，如果程序无法一气呵成，出现许多小错误，那我们会在输入上花费大量的时间。采用输入输出重定向和文件比较可以节省许多时间。

我们先了解一下，输入重定向的命令是”<文件路径”，表示将该文件的内容当作标准输入，输出重定向的命令是”>文件路径”，表示将该文件作为标准输出，该文件事先可以是不存在的。

假设这次的文件也是叫main.c，储存在D:\C_Programming文件夹中，我们只需要像上次那样进入D:\C_Programming，然后输入main <in.txt >out.txt，（假设in.txt和ans.txt也存放在该目录下）便可以将in.txt中的数据代替键盘进行输入，并输出到out.txt中。为什么不输出到屏幕上就好呢？

Windows 命令行中有FC命令，用来比较文本文件是否一样。我们可以用FC out.txt ans.txt比较你的输出与老师的标准答案是否一致，若不一致，屏幕会显示不一致的行，若一致，则会告诉你“找不到差异”或不输出任何东西。如果输出文件是一个 100 行甚至更多数据的文件（比如要把 100000 个数据排序并输出），那这个方法可以带来许多的方便。

§8.3 随机函数

真正意义上的随机数（或者随机事件）在某次产生过程中是按照实验过程中表现的分布概率随机产生的，其结果是不可预测的，是不可见的。而计算机中的随机函数是按照一定算法模拟产生的，其结果是确定的，是可见的。我们可以这样认为这个可预见的结果其出现的概率是 100%。所以用计算机随机函数所产生的“随机数”并不随机，是伪随机数。

在 C 语言里，我们可以调用rand()来生成随机数，为了生成 0 到 n 之间的随机数，我们可以用rand()%(n+1)，因为对n+1取余后，余数不可能超过n了。取 0 至 9 的随机数，便可以用rand()%10来获得。为了保证“随机数”的“随机性”，一般使用系统时间来做种子，这样的话每次运行的种子是不同的。我们调用库函数srand(int)来设置种子，调用time(NULL)来获取当前时间（一个表示从 1970-1-1 到现在经过的秒数的整数）。rand和srand函数需要#include <stdlib.h>;，而time函数，需要#include <time.h>。下面我们为小学生写一个随机生成两个数乘法试题的程序：

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAXN 10
int main(){
    int i, j, k;
    int n, c;
    srand((unsigned)time(NULL));/*设置随机数种子*/
    for(n=0;n<10;n++){
        i=rand()%MAXN;/*产生乘数*/
        j=rand()%MAXN;/*产生乘数*/
        printf("%d*%d=",i,j);
        scanf("%d",&k);/*输入回答*/
        if(k==i*j){/*判断对错*/
            c++;
```

```
16              printf("恭喜你答对了！\n");
17          }
18          else{
19              printf("答错了\n");
20          }
21      }
22      printf("你总共答对了%d道题",c);
23      return 0;
24  }
```

例 8.2 随机生成两个数的乘法试题

同学们可以进一步修改程序，使之可以根据统计答对的题目数量、连续答对的数量、连续答错的次数等等，据此给出不同的鼓励，也可以修改程序，使得用户可以分别设置两个数的上限、下限，这样就可以做一位数跟两位数的乘法等练习……相信大家还有更好玩的想法。

每日一练

(1) 实现并运行例8.2。

第九天　字符串

程序经常要处理字符序列，字符序列最简单的存储方法是用字符数组存储。字符数组就是元素是字符型的数组。对字符序列进行组织处理有多种方法。例如，另外用一个整数来表示这个序列的字符个数。一种更为简便的方法是在字符序列的最后加上一个特殊字符。C 语言就采用在字符序列最后加上一个 ASCII 码为 0 的特殊字符的方法。这个特殊字符用'\0'标记，并称该字符为字符串结束符，而这样的字符序列就称为字符串。

§9.1　字符数组与字符串

§9.1.1　字符串

字符串是由一定数量的字符（字母、数字、符号、转义字符、以及空格等）组成的字符序列。编程中经常把这些字符作为一个整体进行操作。字符串常量是用双引号（""）括起来的 0 个或者多个字符组成的序列，每个字符串尾自动加一个'\0'作为字符串结束标志。例如："China"，"a"，"123"，""，其中""是由 0 个字符组成的空字符串。

§9.1.2　字符数组

与其它类型的数组类似，字符数组其实就是数组元素为字符的数组，定义形式与其它数组一样，例如下面语句定义了一个数组名为str的字符数组，整个数组能存放 6 个字符：

```
char str[6];
```

§9.1.3　字符数组的初始化

除了跟数组的赋初值，字符数组还可以用字符串常量赋初值，例如：

```
char str [6]={"China"};
char str [6]="China";/*可以省略花括号*/
char str [ ]="China"; /*全部赋值，则可以省略数组大小*/
```

§9.1.4　字符串的输出

除了以数组元素的单个字符方式的输出，字符串还可以整体的方式输出（假设str="China"）：

```
printf("%s",str);
```

输出结果为：

```
China
```

§9.1.5 字符数组的输入

除了以数组元素的单个字符方式的输入，字符数组还可以使用字符串的方式输入：

```
scanf("%s", str);                    /*注意输入项中数组名不需要加上&*/
```

但是特别要注意，scanf 使用"%s"格式时是把空白类字符看做是字符串结束的标志，所以输入内容中不能含有空白类字符。例如：输入

```
China
```

则字符数组中存储 6 个字符：'C'，'h'，'i'，'n'，'a'，'\0'。

§9.2 字符串库函数

仅仅是通过之前所介绍的内容要对字符串进行操作是比较麻烦的。其实字符串也有相应的库函数（被包含在头文件string.h中），这些库函数的存在使得处理字符串的过程得到简化。

(1) 字符串输入与输出函数gets()和puts()：函数调用gets(string)可以实现从终端输入字符序列到字符数组string[]中，以回车作为输入结束，函数会将回车符'\n'转换为字符串结束符'\0'存储；函数调用puts(string)可以实现string字符串输出到终端，并将'\0'转化为'\n'输出。相当于printf("%s\n",string)。

(2) 字符串长度函数strlen():string length 缩写。函数调用strlen(string)返回字符串string中的有效字符（不包括字符串结束符'\0'）个数。

(3) 字符串比较函数strcmp()：string compare 缩写。函数调用strcmp(string1，string2)可以实现比较两个字符串。因为字符串不能使用“==”和“!=”运算，所以可以使用这个函数实现对字符串的比较。比较时，从左到右依次对逐个字符（ASCII 码）进行比较，直至出现不同的字符或遇到字符串结束符'\0'为止。如果全部字符相同则返回0值，如果不同则以第一个不相同的字符比较结果为准，若string1的相应字符 ASCII 码大于string2的相应字符，函数返回一个正整数，反之返回一个负整数。

(4) 字符串复制函数strcpy()：string copy 缩写。函数调用strcpy(string1，string2)可以实现将字符串string2的所有字符（包括字符串结束符'\0'）复制到string1中（当然string1要足够大，以便可以容纳下string2字符串中的所有字符），但只对string1中的相应位置进行改变，string1在复制string2时未涉及的位置会保持原值不变。string1不能是字符串常量，可以为字符数组，string2可以为字符串常量。

(5) 字符串连接函数strcat()：string catenate 缩写。函数调用strcat(string1，string2)可以实现将字符串string2复制到字符串string1的后面，所以string1应该足够大以便接纳string2中的所有内容。连接之前string1和string2的末尾均有'\0'，连接其实是去掉string1末尾的'\0'，把string2的内容（包括'\0'）接在string1后面。string1同样不能是字符串常量，可以为字符数组，string2可以为字符串常量。

以上就是一些常用的基本的字符串库函数，其它的字符串库函数，读者可以自行到头文件中查看。

§9.3 例子：字符替换

把字符串中的某个特定的字符X替换为另外一个字符Y。

解题思路：遍历整个字符串，若字符等于X，则把它赋值为Y。例如：将字符串"I love B language." 中的'B'字母替换为'C'，则结果为"I love C language."

代码如下：

```
#include<stdio.h>
#include<string.h>
int main(){
    int i;
    char str[1024], X, Y;/*str为需要进行字符替换原句*/
    printf("Please input a string: ");
    gets(str); /*输入一行字符*/
    printf("Please input character X and Y: ");
    scanf("%c %c", &X, &Y);
    for(i=0; str[i]!='\0'; i++)            /*遍历整个字符串*/
        if(str[i]==X) str[i]=Y;      /*找出需要替换字符并替换*/
    puts(str);                           /*输出结果*/
    return 0;
}
```

例 9.1 字符替换

同学们可以思考一下，如果原句中有多个字符与想要替换的字符相同，而只想要替换第一个匹配的字符，该怎么修改程序？

§9.4 例子：子串查找

在字符串str中找出子串substr所在的位置。

解题思路：遍历整个字符串，若字符串str从第i个位置开始与子串substr的序列相同，则跳出循环。例如：在字符串"I love B language." 中查找子串"language"，则结果为第 9 个字符处。

代码如下：

```
#include<stdio.h>
#include<string.h>
int main()
{
    int i, k;
    char str[]="I love C language."; /*需要进行子串寻找原句*/
    char substr[]="language";  /*所需查找的子串*/

    for(i=0; str[i]!='\0'; i++){
        for(k=0; substr[k]!='\0'; k++)/*查找与子串第一个字符相同的地方，
            依次比较后续字符*/
            if(str[i+k] != substr[k]) break;
        if(substr[k]=='\0') break; /*如果后续字符都相同则跳出循环*/
    }
```

```
14      if(str[i]=='\0') printf("不存在所寻找的子串");
15      else printf("所找的子串首字符在原句的第%d个字符处\n",i);
16      return 0;
17  }
```

例 9.2 寻找子串

同学们可以思考一下，如果原句中有多个子串，如何才能把它们都找出来？如何进行子串的替换？

每日一练

(1) 实现并运行例9.1。

(2) 实现并运行例9.2。

第十天　函数

在用 C 语言开发程序时，开发者首先确定程序的主要功能，然后从主要功能出发，将复杂功能逐层分解成简单的子功能。最后，把完成独立功能的程序编写成函数。当程序需要实现函数功能时，就可以简单地调用函数来完成。

§10.1　定义函数

C 语言中用到的所有的函数，必须“先定义，后使用”，如果事先不定义，编译系统就不知道此函数要实现什么功能。函数定义应包括以下内容：**函数名字，函数的返回值类型**以及**函数参数的名字和类型**。定义函数的一般形式为

```
返回值类型名　函数名(形式参数表列)
{
    函数体
}
```

其中，函数的命名规则与一般的标识符相同，但最好能反映其功能。函数体包括声明部分和语句部分。

```
1 int max(int x, int y){
2     int z;
3     if(x > y) z = x;
4     else z = y;
5     return z;
6 }
```

例 10.1　求两个数中的较大数

§10.2　函数调用

按在程序中出现的形式和位置来分，函数调用有函数调用语句，函数表达式调用和函数返回值作为函数这三种调用方式。

(1) **函数调用语句：**如printf_star();，这时不要求函数带返回值，只要求完成一定操作。

(2) **函数表达式：**如c=max(a,b);。max(a,b)是一次函数调用，是赋值表达式中的一部分。这时要求函数返回一个确定的值以参加表达式的运算。

(3) **函数返回值作为参数：**如printf("%d",max(a,b));是把max(a,b)的返回值作为printf函数的一个参数。

§10.3 全局变量与局部变量

当一个程序中包含两个或多个函数，分别在各个函数中定义变量。这就有一个问题：在一个函数中定义的变量，在其它函数中能否被引用？在不同位置定义的变量，在什么范围内有效？这就是变量的作用域问题。

§10.3.1 局部变量

在一个函数内部定义的变量只在本函数范围内有效，函数以外不能引用，这种变量称为局部变量。注意：

(1) 不同函数中可以使用同名的变量，它们代表不同对象，互不干扰。

(2) 形式参数也是局部变量，其它函数可以调用该函数，但不能直接引用该函数的形参。

(3) 函数内部，可在复合语句（即用大括号括起来的一段代码）中定义变量，这些变量只在本复合语句中有效。

§10.3.2 全局变量

在函数之外定义的变量称为外部变量，外部变量是全局变量。全局变量可以为本文件中其它函数所共用。它的有效范围为从定义变量的位置开始到本源文件结束。注意：

(1) 同一文件中的所有函数都能引用全局变量的值，因此如果在一个函数中改变了全局变量的值，就能影响到其它函数中全局变量的值。

(2) 在同一个源文件中，全局变量与局部变量同名时，在局部变量的作用范围内，局部变量有效，全局变量被“屏蔽”，即它不起作用。

§10.4 例子：二分法解方程

二分法是一种方程式近似根的算法。

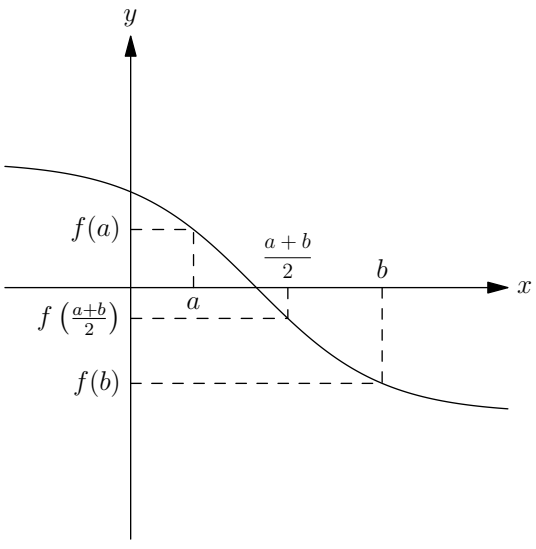

图 10.1 二分法解方程

解题思路：对于方程 $f(x)=0$，我们知道，若 $f(a)f(b)<0$，则区间 $[a,b]$ 内一定包含着方程式的根。因此，可根据以下步骤进行：

(1) 先要找出一个区间 $[a,b]$，使得 $f(a)$ 与 $f(b)$ 异号。 根据零点存在定理，这个区间内一定包含着方程式的根。

(2) 求该区间的中点 $m=\dfrac{a+b}{2}$，并找出 $f(m)$ 的值。

(3) 若 $f(m)$ 与 $f(a)$ 符号相同，则取 $[m,b]$ 为新的区间，否则取 $[a,m]$ 为新的区间。

(4) 重复 (3) 和 (4)，直到得到理想的精确度为止。

接下来我们来看一个具体的例子，求 $x^3-x-1=0$ 的根。程序代码如下：

```c
#include<stdio.h>
#include<math.h>
#define EPS 1.0e-6

double f(double x){
    return (x*x*x-x-1);
}

double binaryRoot(double a,double_t b)
{
    double m;
    while(1){
        m=(a+b)/2;
        if(fabs(f(m)) < EPS) break;/*若精度达到要求，跳出循环*/
        if(f(a)*f(m)<0) /*若f(a)*f(x)<0,则根在区间的左半部分*/
            b=m;
        else if(f(a)*f(m)>0) /*否则根在区间的左半部分*/
            a=m;
    }
    return (a+b)/2; /*取最后的小区间中点作为根的近似值*/
}

int main()
{
    double a, b;
    printf("请输入初始a,b=\n");
    while(1){
        scanf("%lf,%lf",&a,&b);/*输入判断的初始区间*/
        if(f(a) * f(b) > 0) /*判断是否符合二分法使用的条件*/
            printf("二分法不可用，请重新输入：\n");
        else break;
    }
    printf("\n方程 x*x*x-x-1＝0 的一个根是：%f\n",binaryRoot(a,b));
    return 0;
```

```
35 }
```

例 10.2 二分法解方程

程序运行后，输入1，5，则输出

```
1.324718
```

同学们想想，有没有更快一些的方法来加快答案搜索的过程？

每日一练

(1) 实现并运行例10.2。

结束语

在编写程序设计先修课讲义的过程中，六位技术科学大类 15 级的学生参与了讲义的编写。他们有的在中学的时候接触了程序设计，进入大学后学得很快；有的完全是零基础，但经过半年的学习，也掌握了程序设计的基本原理和能力。希望他们的学习经验能给同学们以鼓励。

学弟学妹首先祝贺你们要开始属于自己的编程生活了，作为一个曾经零基础的学姐，没有经验之谈，只希望你们相信，只要用心体会练习，一定可以掌握好这项技能，并在无数次痛苦的出错，改错，重新认识的过程中体会到快乐。学习 C 语言就像是教计算机这样一个小孩子说话，你要让它懂你的意思，要有耐心。平时多上机操作，理论细节理解清楚。祝福你们会在学习的过程中感受到 C 语言的魅力，加油！

（技术科学 15 级 刘婧源）

比尔盖茨说：“编程应该成为 21 世纪人人必备的技能。”现代社会，编程已经融入了生活。学习编程不仅仅有利于思考，也将有利于生活的方方面面。愿这本书能激起你对编程的兴趣，带你走入色彩缤纷的编程世界！

（技术科学 15 级 黄尧）

C 语言的学习可能一开始充满艰辛，但是当你自己用这个平台做出哪怕很小的一个程序，你都会充满成就感。遇到自己不懂的问题，多请教老师，多看书，别放弃，相信你会有很大收获。Are you ready? Let’s say hello to a new world!

（技术科学 15 级 刘宇轩）

亲爱的学弟学妹，不要畏惧未知的天地，勇于探索才会打开新世界的大门。相信在亲手敲击代码的过程中你们会真正对 C 语言产生兴趣，爱上有趣的程序设计。
（技术科学 15 级 赵婧尧）

在这个信息技术已经将各个行业串联起来的时代、在这个人工智能即将在各个领域遍地开花的时代，程序设计语言基本上成为了大家的必修课。C 语言，这个最为经典的程序设计语言，是很多人叩开计算机科学大门的初次体验；而一本简洁的先修课教材，便可以成为大家在假期有限的时间里迅速入门 C 语言的工具。希望大家能够从中有所收获，以此为大学期间学习计算机科学相关知识打下坚实的基础。
（技术科学 15 级 郑溯）

（技术科学 15 级 章俊鑫）